Click2Enforce: โปรแกรมส่วนขยายของเบราว์เซอร์เพื่อป้องกันการโจมตีจากการเปลือยเอสเอสแอล

Click2Enforce: a Browser Extension to Protect against SSL Stripping Attacks

อภิรักษ์ ทูลธรรม (Apirak Tooltham)* และ สมนึก พ่วงพรพิทักษ์ (Somnuk Puangpronpitag)*

บทคัดย่อ

การโจมตี HTTPS (SSL on HTTP) ด้วยเทคนิคการเปลือยเอสเอสแอลถูกนำเสนอตั้งแต่ปี ค.ศ. 2009 จนถึงเวลานี้ เทคนิคดังกล่าวยังคงเป็นภัยคุกคามร้ายแรงต่อการใช้งานโปรแกรมประยุกต์เว็บ เช่น อี-คอมเมิร์ซ ธนาคารออนไลน์ เว็บเมล และเว็บไซต์อื่นๆ อยู่ ซึ่งมีการศึกษาหลายครั้งก่อนหน้านี้ได้ทำการประเมินและแก้ไขปัญหาดังกล่าว แต่อย่างไรก็ตามแนวทางการแก้ไขปัญหาเกือบทั้งหมดยังคงขาดประสิทธิภาพ บางวิธีก็มีความซับซ้อนเกินไปสำหรับผู้ใช้ ดังนั้นงานวิจัยนี้จึงได้นำเสนอ วิธีการแก้ไขปัญหารูปแบบใหม่ คือ Click2Enforce ซึ่งเป็นโปรแกรมส่วนขยายที่ทำงานบนเบราว์เซอร์ จากการทดลองพบว่า Click2Enforce สามารถทำงานได้อย่างมีประสิทธิภาพ และมีความเป็นมิตรกับผู้ใช้ในการแก้ปัญหาการโจมตี

คำสำคัญ: การโจมตีโปรแกรมประยุกต์เว็บ การเปลือยเอสเอสแอล การโจมตีแบบแทรกกลางการสื่อสาร เอชทีทีพี เอสเอสแอล

Abstract

The SSL stripping technique has been proposed since 2009 to attack HTTPS (SSL on HTTP). Till now, this technique is still a serious threat to web applications (such as e-commerce, online banking, webmail and so on). Several previous studies have done to evaluate and fix the problems. However, almost all previous solutions are ineffective; some are too complex for users to collaborate. Hence, this paper proposes a new solution as a browser extension program (named Click2Enforce). From the experiments, Click2Enforce can be effective and very user friendly in tackling the problem.

Keyword: Web Application Attack, SSL Stripping, Man in the Middle Attack, HTTPS, SSL.

1. บทนำ

การโจมตีโปรแกรมประยุกต์เว็บ (Web Application) ด้วยเทคนิคการเปลือยเอสเอสแอล (SSL Stripping Attack) เป็นภัยคุกคามที่มีความน่าสะพรึงกลัว เนื่องมาจากการโจมตีไม่ได้ถูกจำกัดบนแค่แพลตฟอร์มที่เป็น PC Desktop หรือ Notebook เท่านั้น แต่สามารถกระทำการโจมตีไปยังอุปกรณ์อื่น อย่างเช่น Tablet และ Smartphone ได้

SSL Stripping Attack เป็นปัญหาที่เกิดคดีความและความเสียหายขึ้นจริงบนระบบ Online Banking ทั่วโลก ซึ่งพบว่าแม้ผู้ให้บริการจะนำเอาระบบยืนยันตัวตนแบบสองปัจจัย (Two-factor Authentications) อย่าง SMS One-time Password มาใช้ร่วมกับรหัสผ่านหลักเพื่อป้องกันการโจมตี แต่ก็ยังพบว่ามีการนำเทคนิคที่อาศัย Trojan Horse ชื่อ Zeus หรือ Zbot [1] มาใช้ร่วมกับ SSL Stripping Attack ในการจู่โจมระบบจนประสบความสำเร็จ โดยมีกรณีที่แฮ็กเกอร์ทำการโจมตีระบบ Online Banking ของธนาคารพาณิชย์แห่งหนึ่งในประเทศไทยด้วยเทคนิคดังกล่าวมาแล้ว

จากงานวิจัยที่ผ่านมา ซึ่งทำการศึกษาและประเมินถึงปัญหาการถูกจู่โจมด้วย SSL Stripping Attack นั้น ได้เผยให้เห็นถึงความจริงที่น่ากลัวว่า การโจมตีด้วยเทคนิควิธี

* คณะวิทยาการสารสนเทศ มหาวิทยาลัยมหาสารคาม

ดังกล่าวนี้ แม้ว่าโปรแกรมประยุกต์เว็บเหล่านั้นจะได้รับการปกป้องด้วย SSL และใช้เทคนิคการป้องกันรูปแบบอื่นอย่างเช่นการเข้ารหัสข้อมูล (Message Encryption) เข้าร่วมก็ยังอาจเกิดปัญหาการโจมตีขึ้นได้ หรือแม้แต่รูปแบบของการใช้งาน SSL ในลักษณะต่าง ๆ เช่น การใช้ SSL ตลอดขบวนการรับส่งข้อมูล การใช้ SSL เฉพาะช่วงเวลา Login เข้าสู่ระบบ หรือการใช้ SSL เพื่อรับส่งข้อมูลอยู่เบื้องหลัง รวมถึงความแตกต่างทางด้านแพลตฟอร์มของฮาร์ดแวร์ เว็บเบราว์เซอร์ และแม้กระทั่งระบบปฏิบัติการ ก็ไม่อาจสกัดกั้นพลังจู่โจมของเทคนิคการโจมตีรูปแบบนี้ได้ อีกทั้งปัญหาการโจมตีด้วย SSL Stripping Attack ที่เว็บเบราว์เซอร์ไม่สามารถตรวจสอบและแจ้งเตือนการถูกโจมตีได้ ซึ่งปัจจุบัน (มีนาคม ปี ค.ศ. 2013) การโจมตีด้วยเทคนิควิธีดังกล่าวยังคงเป็นปัญหาวิกฤติต่อระบบให้บริการของโปรแกรมประยุกต์เว็บที่ต้องได้รับการแก้ไขอย่างเร่งด่วน

ที่ผ่านมามีการศึกษาและเสนอวิธีแก้ไขปัญหานี้มาหลายครั้ง แต่วิธีการที่ถูกเสนอก่อนหน้านี้ยังมีข้อบกพร่องอยู่ เช่น ความยุ่งยากในการใช้งาน ไม่สามารถปรับระบบป้องกันเพื่อให้ใช้งานกับโปรแกรมประยุกต์เว็บเดิมได้ การไม่รองรับกับแพลตฟอร์มที่หลากหลาย และขาดประสิทธิภาพการป้องกัน

งานวิจัยนี้จึงเสนอวิธีการป้องกันโปรแกรมประยุกต์เว็บจากการโจมตีด้วย SSL Stripping Attack รูปแบบใหม่ที่ใช้งานง่าย และมีประสิทธิภาพในการป้องกันที่ดีขึ้น

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 Secure Socket Layer

Secure Socket Layer Protocol (SSL) หรือมาตรฐานของ IETF ใน RFC 5246 คือ Transport Layer Security (TLS) [2] เป็นโพรโทคอลที่ใช้เข้ารหัสการรับส่งข้อมูล รวมถึงการพิสูจน์ตัวตนในการสื่อสารระหว่างเซิร์ฟเวอร์และไคลเอนท์ ทำให้การสื่อสารผ่านเว็บไซต์มีความปลอดภัยมากยิ่งขึ้น ซึ่ง SSL ทำงานอยู่ในระดับ Transport Layer จึงไม่ถูกจำกัดการใช้เฉพาะในโปรแกรมประยุกต์เว็บเท่านั้น แต่ในโพรโทคอลอื่นๆ อย่าง IMAP, POP3, LDAP, Telnet (SSH) และ FTP ก็สามารถใช้ SSL เพื่อสร้างความปลอดภัยให้แก่การบริการได้

2.2 SSL Stripping Attack

การโจมตีด้วยการเปลือยเอสเอสแอล (SSL Stripping Attack) เป็นวิธีที่ถูกเสนอขึ้นเพื่อโจมตีโปรแกรมประยุกต์เว็บที่ทำงานบน HTTPS โดยอาศัยช่องโหว่ที่เว็บเบราว์เซอร์ไม่สามารถตรวจสอบและกำหนดรูปแบบการสื่อสารบน HTTPS กับเว็บเซิร์ฟเวอร์ ถูกสาธิตครั้งแรกใน Blackhat Conference เมื่อปี ค.ศ. 2009 โดย Marlinspike [3] โดยใช้ SSL Strip เป็นโปรแกรมจู่โจม ถูกพัฒนาจาก Python Script ซึ่งติดตั้งไว้ใน Backtrack Linux ที่กลุ่มผู้เชี่ยวทางด้านระบบเครือข่ายนำไปใช้ในงานด้านการทดสอบและประเมินความมั่นคงของระบบ

การทำงานของ SSL Stripping Attack นั้น อาศัยการโจมตีแบบแทรกกลางการสื่อสาร (Man in the Middle: MITM) ร่วมกับการเปลือย SSL ควบคู่กัน เมื่อโจมตีโปรแกรมประยุกต์เว็บที่ทำงานบน HTTPS แล้ว เว็บเบราว์เซอร์ของเครื่องเป้าหมายจะถูกเปลือย SSL ออก และถูกบังคับให้ใช้ HTTP ในการสื่อสารแทน ส่งผลให้ข้อมูลที่เคยถูกปกป้องโดย SSL ถูกดักจับได้

2.3 งานวิจัยที่เกี่ยวข้อง

อภิรักษ์ และคณะ [4] ได้ทำการประเมินปัญหาของการจู่โจมด้วยการเปลือยเอสเอสแอล โดยทำการทดสอบบนแพลตฟอร์มที่มีความหลากหลายของฮาร์ดแวร์ เว็บเบราว์เซอร์ และระบบปฏิบัติการ พบว่าแม้โปรแกรมประยุกต์เว็บจะถูกใช้งานบนแพลตฟอร์มที่แตกต่างกัน หรือมีการใช้ SSL ร่วมกับการป้องกันรูปแบบอื่นใดก็ตาม ระบบการให้บริการของโปรแกรมประยุกต์เว็บเหล่านั้นก็ยังสามารถถูกโจมตีด้วยวิธี SSL Stripping Attack ได้

ในอดีตที่ผ่านมามีงานวิจัยที่ได้ทำการศึกษาและออกแบบแนวทางการแก้ไขปัญหา SSL Stripping Attack ดังต่อไปนี้

Nikiforakis และคณะ [5] ได้เสนอ HProxy เพื่อป้องกัน MITM และ SSL Stripping Attack โดยติดตั้งระบบจัดเก็บประวัติการใช้งานเว็บไซต์เพื่อตรวจสอบว่ากำลังใช้งานอยู่บน HTTPS ที่ปลอดภัยหรือไม่ โดยแสดงผลแจ้งเตือนบนเว็บเบราว์เซอร์ และต้องติดตั้ง Client-side Proxy และ JavaScript ที่กำหนดการทำงานไว้บนเว็บเซิร์ฟเวอร์ด้วย ระบบนี้สามารถตรวจสอบการโจมตีได้เท่านั้น ซึ่งบางครั้งพบว่าระบบไม่ทำการแจ้งเตือนในขณะที่เว็บไซต์กำลังถูกโจมตีอยู่ (False Negative)

Fung และคณะ [6] ได้เสนอ SSLock เป็นข้อเสนอวิธีการบังคับใช้ SSL โดยแบ่ง Domain Name ที่ใช้งาน SSL และกำหนดรูปแบบการตอบกลับ HTTP Header ชื่อ SSLock-

Candidates พร้อมกับ JavaScript ที่ใช้อ่าน HTTP Header ซึ่งในเว็บเบราว์เซอร์จะเปลี่ยนเป็น Domain Name ใหม่ที่อยู่บน HTTPS และส่งคำขอไปที่เว็บเซิร์ฟเวอร์ แต่มีข้อเสียในด้านมาตรฐานในการพัฒนาระบบ และทำได้เพียงแค่การตรวจสอบ

Fung และคณะ [7] ได้เสนอ HTTPSLock เป็นข้อเสนอการบังคับใช้งาน HTTPS ที่มีการใช้งานใบรับรองที่ถูกต้อง (Valid Certificate) ด้วย JavaScript หากเว็บไซต์ที่ใช้งานอยู่มีสถานะการใช้งานใบรับรองที่ไม่ถูกต้อง (Invalid Certificate) ระบบจะไม่อนุญาตให้ใช้งานได้ และกรณีมีการใช้งานใบรับรองที่ถูกต้อง แต่ขณะใช้งานพบว่าอยู่บน HTTP ก็จะแสดงผลว่าการใช้งานมีความเสี่ยง ซึ่งวิธีนี้ทำได้แค่ตรวจสอบการโจมตี

Shin และ Lopes [8] ได้เสนอ SSLight เพื่อใช้การสังเกตในการตรวจสอบ พัฒนาเป็น Browser Extension บน Google Chrome ซึ่งแสดงการแจ้งเตือนด้วยสีใน Login Form โดยสีแดงจะหมายถึงไม่ปลอดภัย สีเหลืองหมายถึงระมัดระวัง และสีเขียวหมายถึงปลอดภัย แต่ระบบทำได้เพียงแค่การแจ้งเตือนการโจมตีเท่านั้น อีกทั้งไม่รองรับกับทุกเว็บเบราว์เซอร์

Hodges และคณะ [9] ได้เสนอ HTTP Strict Transport Security (HSTS) ซึ่ง IETF กำหนดเป็นมาตรฐาน RFC 6797 เพื่อแก้ไขปัญหา SSL Stripping Attack โดยกำหนดให้เว็บเบราว์เซอร์ทำการบังคับใช้ HTTPS ตามที่ถูกระบบกำหนดไว้ แต่ระบบนี้รองรับเพียง Google Chrome และ Mozilla Firefox เท่านั้น และจำกัดเฉพาะเว็บไซต์ที่อยู่ใน HSTS List จึงทำให้การใช้งานเว็บไซต์อื่นไม่สะดวก เพราะต้องมากำหนดค่าใหม่ด้วยตนเอง

ณัฐวุฒิ และคณะ [10] ได้เสนอการแก้ไขปัญหาการโจมตีเว็บไซต์บน HTTPS โดยเก็บรายชื่อเว็บไซต์ไว้ใน Bookmark ของเว็บเบราว์เซอร์ เพื่อแก้ปัญหาความไม่สะดวกในใช้งานและปัญหาการไม่รองรับทุกเว็บเบราว์เซอร์ แต่มีข้อเสียคือความยุ่งยากในการบันทึก URL ลงใน Bookmark ใหม่ทุกครั้ง เมื่อมีการใช้งานเว็บไซต์ที่ไม่เคยถูกบันทึก และหากใช้งานเว็บเบราว์เซอร์หลายโปรแกรมก็จะต้องทำการเพิ่มข้อมูลลงในทุกโปรแกรมด้วย

Puangpronpitag และคณะ [11] ได้เสนอ ISAN-HTTPS Enforcer ที่พัฒนาโดย JavaScript ซึ่งสามารถทำงานได้กับทุกเว็บเบราว์เซอร์โดยไม่ต้องปรับเปลี่ยนหรือติดตั้ง Plug-in เพิ่ม สามารถใช้งานได้กับ Platform ที่มีความหลากหลาย แต่มีข้อเสียที่ Overhead ของ Response ที่ใช้ประมวลผล JavaScript และ SSL Stripping Attack สามารถแก้ไข Code เพื่อทำการปลด Tag ที่ใช้ในการป้องกันออกได้

ACIS Professional Center [12] พัฒนา SSLStrip Guard ซึ่งเป็น Application บน Android และ IOS เพื่อตรวจสอบการถูก SSL Stripping Attack ในเครือข่ายไร้สาย โดยติดต่อไปยังเซิร์ฟเวอร์ SSL ที่กำหนด ไว้เพื่อตรวจสอบการสื่อสารว่าอยู่บน HTTPS หรือไม่ แล้วแจ้งเตือนผลให้ทราบ แต่เป็นแค่การตรวจสอบการโจมตี ซึ่งจะต้องตรวจสอบก่อนการใช้งานทุกครั้ง และการทำงานมีการเรียกใช้งาน HTML และ JavaScript ซึ่งในการโจมตีหากแฮ็กเกอร์ปรับแก้ Python Script ของ SSL Strip เพียงเล็กน้อยก็สามารถ Bypass การป้องกันได้

จากแนวทางการแก้ไขปัญหาต่างๆ ที่ถูกนำเสนอมาพบว่า HProxy, SSLock, HTTPSLock, SSLight และ SSLStrip-Guard ทำได้เพียงแค่การตรวจสอบแล้วแจ้งเตือนการถูกโจมตีให้กับผู้ใช้ทราบ โดยมีเพียง Bookmark, HSTS และ ISAN-HTTPS Enforcer เท่านั้นที่สามารถป้องกันการโจมตี SSL ได้ แต่วิธีดังกล่าวเหล่านี้ต่างยังมีข้อบกพร่องอยู่ เช่น ความยุ่งยากในการใช้งาน การป้องกันไม่ได้ผลจริง การป้องกันทำได้ในแค่ระดับของ Script Kiddies เท่านั้น การป้องกันไม่สามารถปรับใช้งานกับเว็บไซต์เดิมได้ การไม่รองรับกับแพลตฟอร์มอื่นที่หลากหลาย และขาดประสิทธิภาพการป้องกัน

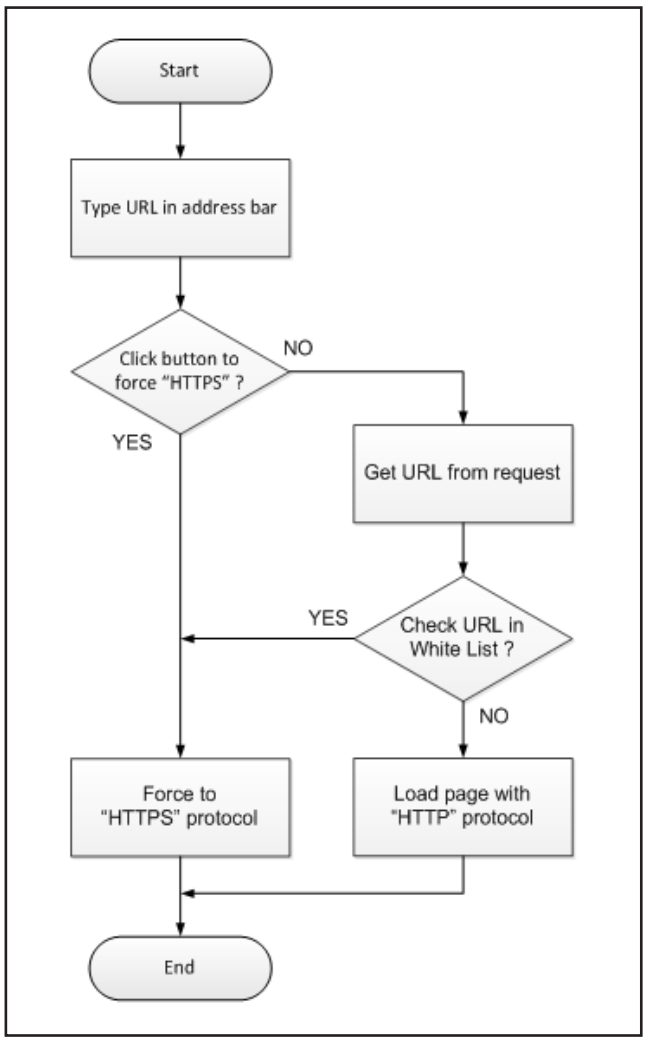
3. ระบบป้องกันการโจมตี SSL รูปแบบใหม่

การทำงานของระบบโปรแกรมประยุกต์เว็บที่มีความปลอดภัยนั้น ระบบจำเป็นจะต้องมีการสื่อสารอยู่บนโพรโทคอล HTTPS เนื่องจากสามารถป้องกันการโจมตีได้ในหลากหลายรูปแบบ ทั้งการโจมตีแบบแทรกกลางการสื่อสาร การโจมตีด้วยเทคนิค SSL Sniffing Attack และ SSL Stripping Attack

งานวิจัยนี้จึงนำเสนอแนวคิดของอัลกอริทึมในการออกแบบระบบป้องกันการโจมตี SSL ซึ่งสามารถนำอัลกอริทึมนี้ไปประยุกต์เพื่อใช้ในการออกแบบและพัฒนาระบบการป้องกันการโจมตี SSL บนเว็บเบราว์เซอร์อื่นได้ต่อไป โดยในการพัฒนาระบบการป้องกันซึ่งเป็นต้นแบบ (Prototype) นี้ ได้เลือก Google Chrome มาเป็นเว็บเบราว์เซอร์ต้นแบบ

3.1 แนวความคิดในการพัฒนาระบบ

ด้วยรูปแบบของ SSL Stripping Attack มีพฤติกรรมการโจมตีในลักษณะของการเปลี่ยนแปลงโพรโทคอลในการสื่อสารจาก HTTPS เป็น HTTP ในงานวิจัยนี้จึงได้ออกแบบระบบป้องกันที่เว็บเบราว์เซอร์สามารถบังคับการใช้โพรโทคอล HTTPS ในการสื่อสาร โดยมีแนวคิด คือ เมื่อผู้ใช้ทำการพิมพ์ URL บน Address Bar ของเว็บเบราว์เซอร์แล้วทำการร้องขอไปยังเว็บเซิร์ฟเวอร์ที่ให้บริการ หากพบว่าโพรโทคอลที่ใช้ในขณะนั้นเป็น HTTPไม่ใช่ HTTPS ที่ปลอดภัย ผู้ใช้สามารถบังคับใช้ HTTPS ได้โดยการคลิกที่ปุ่มไอคอนของระบบป้องกัน หรือกรณีหากยังไม่มีการคลิกปุ่ม แต่ระบบป้องกันตรวจสอบพบว่า URL ดังกล่าวถูกบรรจุอยู่ในรายชื่อเว็บไซต์ที่ต้องการป้องกัน (White List) ระบบป้องกันก็จะบังคับใช้โพรโทคอล HTTPS ให้โดยอัตโนมัติ ดังภาพที่ 1



ภาพที่ 1 *Flow Diagram ของระบบป้องกัน*

3.2 การพัฒนาระบบป้องกันการโจมตี

จากการทำงานของ Flow Diagram ที่เป็นแนวคิดในการออกแบบจึงนำมาสู่อัลกอริทึมของระบบการป้องกันการโจมตี โดยแบ่งขบวนการทำงานออกเป็นสองส่วนสำคัญ คือ ส่วนแรกเป็นการป้องกันการโจมตีโดยผู้ใช้ต้องทำการคลิกที่ปุ่มด้วยตนเอง (Manual Type) และส่วนที่สองจะเป็นการอ่านค่าของ URL แล้วนำมาเปรียบเทียบกับรายชื่อเว็บไซต์ที่ถูกบันทึกเอาไว้ (ซึ่งผู้ใช้สามารถที่จะเพิ่มข้อมูลได้เอง) หากตรงกันก็จะทำการป้องกันให้ทันทีโดยที่ผู้ใช้ไม่ต้องคลิกที่ปุ่ม (Automatic Type) ซึ่งอัลกอริทึมของการทำงานแสดงไว้ดังภาพที่ 2

```
1  url = get(address bar);
2  HTTPS = url.replace('http:','https:');
3  browserAction.onClicked.addListener(function(tab) {
4        HTTPSTab = HTTPS.get(tab.url);
5        tabs.update(tab.id, {url: HTTPSTab});
6  });
7
8  localStorage[installed]
9  badRequest = 0;
10
11 webRequest.onBeforeRequest.addListener(function(details) {
12       if (details.requestId == badRequest) return;
13       if (details.method.toLowerCase() != "get") return;
14       regex = http://(\w?);
15       return { "redirectUrl": details.url.replace(regex, 'https://$1') };
16 }, { "urls": whitelist }, [ "blocking" ]);
17
18 webRequest.onBeforeRequest.addListener(function(details) {
19       if (details.requestId == badRequest) return;
20       if (details.method.toLowerCase() != "get") return;
21       urlWk = details.url;
22       for (var i in patterns) urlWk = urlWk.replace(patterns[i][0], patterns[i][1]);
23       if (urlWk != details.url) return {"redirectUrl":urlWk};},{"urls":["http://*/*"]},["blocking"]);
24
25 webRequest.onBeforeRedirect.addListener(function(details) {
26       if (details.redirectUrl.indexOf('http:') == 0)
27            badRequest = details.requestId;},{'urls': [ 'https://*/*' ] });
```

ภาพที่ 2 *อัลกอริทึมการทำงานของ Click2Enforce*

ด้วยอัลกอริทึมที่เป็นต้นแบบของแนวคิดในการพัฒนา Click2Enforce จึงถูกพัฒนาขึ้นเป็นโปรแกรมส่วนขยายของเว็บเบราว์เซอร์ (Browser Extension) บน Google Chrome ด้วยภาษา HTML 5, JavaScript และ CSS ซึ่งประกอบไปด้วยไฟล์ต่าง ๆ ดังนี้

1) manifest.json ใช้กำหนดรายละเอียดของการพัฒนาและค่าเริ่มต้นของโปรแกรม

2) background.html ใช้เพื่อทำการเรียกใช้งาน JavaScript ในไฟล์ background.js

3) background.js หลังจากถูกเรียกใช้งานแล้วจะคอยทำงานอยู่เบื้องหลัง (Background Process) เพื่อป้องกันระบบ

4) preload.js ใช้กำหนดรูปแบบของ URL และเก็บบันทึกรายชื่อของเว็บไซต์ที่ต้องการทำการป้องกันการโจมตี

5) icon.png ใช้เป็นภาพของปุ่มกดบนเว็บเบราว์เซอร์

หลังจากทำการสร้างไฟล์ที่เกี่ยวข้องทั้งหมดครบถ้วนแล้ว จึงทำการรวมไฟล์เข้าด้วยกัน (Pack Extension) ซึ่งจะได้เป็นไฟล์นามสกุล .crx สำหรับใช้ติดตั้งบนเบราว์เซอร์ต่อไป

ตารางที่ 1 *เว็บไซต์ที่ใช้ประเมินปัญหาการใช้งาน SSL*

Website	URL
ธ.กรุงเทพ	ibanking.bangkokbank.com
ธ.กรุงไทย	www.ktbonline.ktb.co.th
ธ.กรุงศรีอยุธยา	www.krungsrionline.com
ธ.กสิกรไทย	online.kasikornbankgroup.com
ธ.เกียรตินาคิน	ebanking.kiatnakin.co.th
ธ.ซิตี้แบงก์	www.citibank.co.th
ธ.ซีไอเอ็มบี ไทย	www.cimbclicks.in.th/ibkthai
ธ.ทหารไทย	www.tmbdirect.com
ธ.ทิสโก้	www.tisco.co.th
ธ.ไทยพาณิชย์	www.scbeasy.com
ธ.ธนชาต	retailib.thanachartbank.co.th
Citi Group	online.citibank.com
Standard Chartered	online-banking.standard chartered.co.th
Paypal	paypal.com/th
Paysbuy	www.paysbuy.com
Yahoo!mail	login.yahoo.com
Gmail	www.gmail.com
Hotmail	www.hotmail.com
Facebook	www.facebook
Twitter	twitter.com

หมายเหตุ: ด้วยข้อจรรยาบรรณแล้ว ในการทดลองนี้ไม่ได้มีวัตถุประสงค์ในการเจาะเข้าไปในระบบการให้บริการของเว็บไซต์ที่ใช้ในการทดสอบ แต่เป็นเพียงการทดสอบดักจับข้อมูลในระหว่างการสื่อสารเท่านั้น การทดลองจึงไม่ได้ใช้ชื่อผู้ใช้และรหัสผ่านจริง

ภาพที่ 3 *Click2Enforce ที่ถูกติดตั้งบน Google Chrome*

4. การทดสอบระบบป้องกันที่ถูกพัฒนาขึ้น

การทดสอบประสิทธิภาพของระบบ Click2Enforce นั้น จะทำการทดสอบกับกลุ่มตัวอย่างเว็บไซต์จำนวน 20 เว็บไซต์ จากระบบให้บริการ Online Banking, Web Mail และ Social Network ที่เป็นที่นิยมและเป็นเป้าหมายของการโจมตี ดังตารางที่ 1

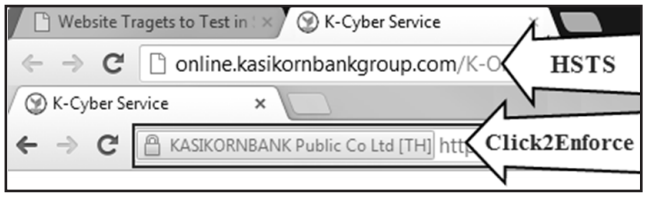
โดยการทดสอบประสิทธิภาพของระบบ Click2Enforce จะทำการเปรียบเทียบกับระบบ HSTS ที่เป็นระบบการป้องกันการโจมตี SSL ซึ่งได้รับการยอมรับในระดับสากล ในการทดสอบนั้นได้กำหนดเกณฑ์การประเมินไว้ 2 ด้าน คือ ประสิทธิภาพการป้องกันการโจมตี และประสิทธิภาพในการใช้งาน

4.1 การทดสอบประสิทธิภาพการป้องกันการโจมตี

การทดสอบโจมตีกลุ่มตัวอย่างเพื่อวัดประสิทธิภาพของ Click2Enforce เทียบกับ HSTS พบว่าระบบที่ถูกพัฒนาขึ้นมีประสิทธิภาพเหนือกว่าระบบ HSTS เนื่องจาก HSTS สามารถทำการป้องกันการโจมตีได้เพียง 5 จากทั้งหมด 20 เว็บไซต์ โดยที่เป็นการป้องกันที่แท้จริงจาก HSTS มีเพียง 3 เว็บไซต์ ส่วนอีก 2 เว็บไซต์นั้น เกิดจากความผิดพลาดของการ Redirection เพื่อกลับไปใช้ HTTPS อีกครั้ง จนเกิดเป็นวงรอบของลูปที่ไม่รู้จบ จนกลายเป็นการปฏิเสธการให้บริการ (Denial of Service: DoS) ในที่สุด ซึ่งผลการทดสอบสรุปไว้ดังตารางที่ 2

ตารางที่ 2 *ผลการทดสอบประสิทธิภาพในการป้องกันการโจมตี*

Solution	Number of sites protected
HSTS	5
Click2Enforce	20



ภาพที่ 4 *ผลการทดสอบการใช้ Click2Enforce และ HSTS*

4.2 การทดสอบประสิทธิภาพในการใช้งาน

การประเมินประสิทธิภาพในการใช้งานนั้น จะใช้วิธี KLM (Keystroke-level Model) ทดสอบด้วยการกำหนดชื่อเว็บไซต์ facebook.com ซึ่งไม่อยู่ใน White List ของ Click2Enforce และ HSTS โดยเริ่มการทดสอบตั้งแต่ขบวนการป้อนข้อมูลในเว็บเบราว์เซอร์เพื่อกำหนดค่าให้กับระบบป้องกัน และสิ้นสุดการทดลองเมื่อเรียกใช้งานเว็บไซต์แล้วระบบดังกล่าวสามารถทำการป้องกันเว็บไซต์ที่กำหนดได้ ผลของการทดสอบแสดงเวลาทั้งหมดที่ใช้ในการดำเนินกิจกรรมต่าง ๆ บนเว็บเบราว์เซอร์ ดังตารางที่ 3

ตารางที่ 3 *ผลการทดสอบประสิทธิภาพในการใช้งานด้วยวิธี KLM*

Solution	Total Time (Second)
HSTS	13.17
Click2Enforce	4.2

จากการทดสอบประสิทธิภาพในการใช้งานด้วยวิธี KLM พบว่าเวลาที่ใช้ใน Click2Enforce ที่ถูกพัฒนาขึ้น น้อยกว่าระบบ HSTS แสดงให้เห็นว่า เมื่อผู้ใช้ต้องการความปลอดภัยในการใช้งานโปรแกรมประยุกต์เว็บโดยใช้วิธีการป้องกันที่นำเสนอในงานวิจัยนี้ จะมีความสะดวก และง่ายในการเข้าถึงการทำงานกว่า

5. สรุปผลการวิจัย

การเสนอวิธีการแก้ไขปัญหา SSL Stripping Attack ในงานวิจัยที่ผ่านมา ส่วนใหญ่เป็นเพียงแค่การตรวจสอบแต่ไม่ใช่การป้องกันการโจมตี ซึ่งพบปัญหาในด้านประสิทธิภาพของการป้องกัน ปัญหาความไม่สะดวกในการใช้งาน หรือปัญหาในการปรับปรุงระบบป้องกันเพื่อให้เข้ากับการทำงานของเว็บไซต์เดิม ในงานวิจัยนี้จึงได้พัฒนาระบบป้องกันการโจมตี SSL โดยเสนอ Click2Enforce ซึ่งถูกออกแบบและพัฒนาในรูปแบบของ Browser Extension บน Google Chrome เพื่อพัฒนาเป็นต้นแบบที่จะนำไปสู่การสร้างระบบการป้องกันการโจมตีบนเว็บเบราว์เซอร์อื่นต่อไป จากผลการทดสอบชี้ให้เห็นว่าระบบป้องกัน Click2Enforce ที่ถูกพัฒนานี้ มีความสะดวกในการใช้และประสิทธิภาพในการป้องกันการโจมตีที่ดีขึ้น

ในอนาคตระบบการป้องกันที่พัฒนานี้จะถูกปรับปรุงให้มีความสามารถในการวิเคราะห์เว็บไซต์ที่ถูกใช้งานไปแล้วว่ามีการใช้ SSL อยู่หรือไม่ หากมีการใช้ SSL อยู่ ระบบจะทำการบันทึกเก็บไว้ในรายชื่อเว็บไซต์ที่เป็น White List เพื่อบังคับใช้ HTTPS โดยอัตโนมัติเมื่อเข้าใช้งานเว็บไซต์ดังกล่าวอีกครั้ง

6. เอกสารอ้างอิง

[1] Kaspersky Lab, “ZeuS on the Hunt.” Available online at http://www.securelist .com/en/analysis/204792107/ZeuS_ on_the_Hunt, 2012.

[2] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2.” *IETF, RFC 5246*, 2008.

[3] M. Marlinspike, “New Tricks For Defeating SSL In Practice.” Available online at https://www.blackhat.com/presentations/bh-dc-09/Marlinspike /BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf, 2012.

[4] A. Tooltham and S. Puangpronpitag. “The Evaluation of the SSL Stripping Attack Problem.” *Proceedings of the National Conference on Computer Information Technologies*, Sakon Nakhon, Thailand, pp. 43-48, 24 Jan, 2013.

[5] N. Nikiforakis, Y. Younan and W. Joosen, “HProxy: client-side detection of SSL stripping attack.” *Proceedings of the 7th international conference on Detection of intrusions*, malware, and vulnerability assessment, Bonn, Germany, pp. 200-218, 8 - 9 Jun. 2010.

[6] A. Fung and K. Cheung, “SSLock: sustaining the trust on entities brought by SSL,” *Proceedings of the 5th ACM Symposium on Information*, Computer and Communications Security, Beijing, China, pp. 204-213, 13 - 16 Apr, 2010.

[7] A. Fung and K. Cheung, "HTTPSLock: Enforcing HTTPS in Unmodified Browsers with Cached Javascript." *Proceedings of the 4th Network and System Security*, Melbourne, Australia, pp. 269-274, 1 - 3 Sep, 2010.

[8] D. Shin and R. Lopes, "An empirical study of visual security cues to prevent the SSL Stripping attack." *Proceedings of the 27th Annual Computer Security Applications Conference*, Orlando, Florida, pp. 287-296, 5 - 9 Dec, 2011.

[9] J. Hodges, C. Jackson and A. Barth, "HTTP Strict Transport Security (HSTS)." *IETF*, *RFC 6797*, Nov, 2012.

[10] N. Sriwiboon and S. Puangpronpitag. "Detection & Protection Mechanisms Against SSL Strip Attacks." *Proceedings of 9th International Joint Conference on Computer Science and Software Engineering*, Bangkok, Thailand, pp. 25-30, 30 May - 1 Jun, 2012.

[11] S. Puangpronpitag and N. Sriwiboon. "Simple and Light-weight HTTPS Enforcement to Protect Against SSL Striping Attack." *Proceedings of 4th International Conference on Computational Intelligence*, Communication Systems and Networks, Phuket, Thailand, pp. 229-234, 24 - 27 Jun, 2012.

[12] ACIS Professional Center, "SSLSTRIPGuard version 1.01." April, 2012.